SPECIFICATION

Electronic Version 1.2.8

Stylesheet Version 1.0

A Circuit and Method for Modeling I/O

Background of the Invention

[0001] This invention claims prority based on Provisional Patent Application Number 60/288,813 filed on May 5, 2001.

[0002] This invention relates to methods and systems used for generating behavioral models used in integrated circuit design. More particularly, the present invention provides for a new behavioral model that provides timing, noise and integrity grid analysis.

[0003] When simulating I/O electrical performance during timing characterization, signal integrity analysis, and power grid integrity analysis, various I/O modeling techniques have been used. At one end of the modeling spectrum are the full netlist models that contain detailed architectural and parasitic information of the I/O. These models provide the highest level of accuracy and can be used for a variety of analysis. A major disadvantage of full netlist models are excessive simulation times that prohibit them from being used at the chip level and non-convergence under certain conditions. At the other end of the spectrum are empirical models for driver delay and IBIS models for signal integrity analysis.

[0004] Empirical models use simple equations or lookup tables for predicting driver delay and output slew rate. The advantage of empirical models is fast simulation time. The disadvantages are poor accuracy under certain conditions, and they are typically limited to timing analysis. For signal integrity analysis IBIS models can be used. The advantage of these models are accurate driver output waveforms across a wide range of loading conditions. The disadvantages are they cannot be used for timing analysis and the models do not predict driver sensitivity to variations in supply voltage, temperature, and input slew rate.

[0005] I/O behavioral modeling in the form of IBIS models has gained wide acceptance in signal integrity analysis. While the IBIS model accurately represents the characteristics of the output pin at three fixed process corners, it does not model driver delay or account for variations in temperature, supply voltages, and input transition rate. The IBIS models used today by various board level simulation tools for signal integrity analysis are behavioral in nature and offer the user and developer of the models several advantages over full-netlist models. First, because IBIS models are behavioral, they contain no proprietary information. This makes it easy to exchange information about I/O characteristics without disclosing intellectual property. Second, behavioral simulation is faster than full-netlist simulation (e.g., Spice) because it uses higher-level abstraction models. What would be prohibitive in terms of simulation time when using full-netlist models can be accomplished in reasonable time with behavioral models.

[0006] The IBIS specification (ANSI/EIA-656-A, "I/O Buffer Information Specification (IBIS) Version 3.2", Sept.1999), presents several techniques for improving model accuracy across a wide range of I/O family types for signal integrity analysis. I/O behavior modeling (e.g. IBIS I/O Buffer Information Specification) have been used by industry in PCB level signal integrity tools such as SpectraQuest from Cadence and XTK from Viewlogic for several years.

Brief Summary of the Invention

[0007] Unfortunately, the IBIS techniques do not model driver delay, pre-drive currents, or n-well decoupling structures which would be required for timing, noise, and power grid integrity analysis. In addition, no techniques are available to the IBIS model developer to account for variations in temperature, supply voltages, and input transition rate. This invention provides a new and powerful behavior model that overcomes the IBIS limitations and is appropriate for merged digital-analog timing, noise, and power grid analysis.

[0008]

A new modeling technique provided by this invention preserves the accuracy of the full netlist model, simulates orders of magnitude faster, and can be used for timing, power grid integrity, and signal integrity analysis over a wide range of supply voltage(s), temperature, input slew rate, and loading conditions. Because of the relative simplicity of this new modeling technique as compared to a full netlist model, non-convergence issues at run time are ameliorated.

[0009] Additional advantages of this invention are 1) the behavioral model provided here captures driver delay, thereby allowing on-chip and off-chip timing to be merged into a seamless interface, 2) I/O current waveforms are readily available for on-chip power grid integrity analysis, and 3) signal integrity analysis at the chip level is available without the need of procuring othre software. Furthermore, the behavioral model provided by this invention can be used to enhance other applications that currently perform package noise analysis using full netlist models which greatly limits the cross sectional size it can analyze at one time. These advantages enable these models, herein referred to as "BIO" (Behavioral I/O) models, to form a seamless interface between on-chip and off-chip timing. They can also provide accurate rail current waveforms for power grid analysis tools and I/O placement tools. Because BIO models have the ability to handle ground bounce and power supply collapse, they can also be used in signal integrity tools.

[0010] In order to accomplish this invention provides a method for modeling the inputs and outputs of integrated circuits, comprising the steps of representing in the model the output characteristics of driver circuits by two types of elements, switching and non-switching; tabulating the output characteristics for each of the elements by applying a DC voltage source on the output of the driver and measuring the current through each element; representing in the model switching elements as a voltage-time controlled resistors by obtaining the product of DC impedance as a function of voltage and a scalar that is a function of time; and embedding in the model equations that are functions of input edge arrival times and cycle time for each scalar type.

[0011]

The circuit used for modeling comprises switching elements connected serially as voltage-time controlled resistors; one of the conductive elements acts to pull voltage up, while the other conductive elements acts to pulls the voltage down; and non-switching elements connected serially as resistors, one representing power structures and the other representing ground clamping structures; each of the switching elements is tied to input stage and both the switching and non-switching elements is tied to an output.

Brief Description of the Drawings

[0012] Figure 1 illustrates an example driver structure used to illustrate the characterization techniques of both switching and non-switching elements.

[0013] Figure 2a illustrates waveforms for characterizing the NFET turn-on characteristics of a driver.

[0014] Figure 2b illustrates characterization of the nfet turn-on and pfet turn-off rates.

[0015] Figure 2c illustrates the characterization of the pfet turn-on and nfet turn-off rates represented as scalars values as a function of time.

[0016] Figure 3 illustrates sensitivity to variable parameters such as temperature for a NFET turn-on scalar.

[0017] Figure 4a illustrates how the DC conductance is obtained.

[0018] Figure 4b illustrates how the transient conductance of a device is obtained.

[0019] Figure 4c illustrates the transient conductance of a device.

[0020] Figure 5a illustrates the determination scalar values as a function of time at some fixed voltage and temperature denoted by Env1.

[0021] Figure 5b illustrates how the scalars values from Figure 5b are represented in tabular format within the BIO model.

[0022] Figure 5c is simple illustration of how these scalar values might change due to variations in voltage, temperature, and input slew rate denoted by Env2.

[0023] Figure 6 illustrates the behavioral model topology provided by this invention.

[0024] Figure 7 illustrates a static timer using the BIO model of this invention.

[0025] Figure 8 illustrates the use of the BIO model in the mixed signal environment.

[0026] Figure 9 illustrates a typical hardware configuration of an information handling/computer system for modeling. integrated circuits in accordance with the invention.

[0027] Figure 10 illustrates an example of signal bearing media.

[0028] Figure 11 illustrates a characterization of the decoupling stage of the model topology of Figure 6.

Detailed Description of the Invention

[0029] The output characteristics of a typical driver are represented by two basic element types: switching and non-switching. Switching elements are functions of both time-varying and non-time-varying parameters, and non-switching elements are functions of non-time-varying parameters only. Examples of switching elements are nfets and pfets. Typically, a switching element is used to represent the composite transient impedance (conductance) behavior of a pull-up or pull-down network that are comprised of multiple FETs and parasitics. Examples of non-switching elements are resistive termination's and ESD structures. ESD impedance (conductance) is essentially a function of device voltage only and can therefore can be treated as a non-switching element. There are, however, ESD structures that can be dynamically switched that would require a switching element to represent its impedance.

[0030] Figure 1 illustrates an example driver structure used to illustrate the characterization techniques of both switching and non-switching elements. These techniques can easily be extended to other I/O structures containing devices that can be classified as either switching or non-switching.

[0031] The first step is to characterize the dc impedance for each element in Figure. 1 at a fixed temperature and supply voltage. This impedance, which we call the dc_base impedance, is tabulated as a function of device voltage. To account for variations in temperature and supply voltages, device impedance can be obtained from the dc_base according to equation (1) below.

[0032] dc_impedance = (1+D0)*dc_base (1)

[0033] The value for D0 is calculated prior to simulation time using an empirical equation which is a function of both temperature and supply voltage. The sensitivity this parameter has to device voltage is small, allowing the use of a simple scalar technique across the full range of dc_base impedance values.

[0034] For each element type, it's dc impedance as a function of the voltage across it is tabulated as shown in Table 1 below:

[0035]

TABLE DC POWER CLAMP

Vpowerclamp	Resistance
0	10000
0 .7	100
5	1

TABLE DC GROUND CLAMP

Vgroundclamp	Resistance
0	10000
0.7	100
5	1

TABLE DC PFET

Vpfet	Resistance
10	40
5	42
0	43

TABLE DC NFET

Vnfet	Resistance
5	40
0	42
-5	43

Table 1

[0036] In addition to dc impedance, for each switching type of element, the turn-on and turn-off rates are characterized by observing the transient impedance during rising and falling output waveforms.

[0037] Figure 2a illustrates waveforms for characterizing the NFET turn-on characteristics of a driver. The transient impedance of the NFET turning on is a function of device voltage and of "local time", that begins at the midpoint of the input transition. Normalizing the transient impedance to the dc impedance (function of device voltage) produces a time-varying scalar representing the turn-on rate of the NFET, independent of the load that was used during characterization. This scalar is unity when the NFET is completely turned on, and very large when the NFET is off. For this same transition and same "local time", another time-varying scalar is obtained for the PFET turn-off. Similarly, a different "local time" for the output-rising transition is used as the basis for PFET turn-on and NFET turn-off scalars. This collection of four scalars is saved in tabular format. Figure 2b illustrates characterization of the nfet turn-on and pfet turn-off rates. Figure 2c illustrates the characterization of the pfet turn-on and nfet turn-off rates represented as scalar values as a function of time.

[0038] In order to enable behavioral simulation of periodic waveforms, the "local times" are made periodic through their definitions as functions of periodic rising and falling input edge arrival times. The time indexing of each scalar is unique. For example, the zero time reference for the nfet_on and pfet_off scalars corresponds to the time a falling input wave crosses it's midpoint while the nfet_off and pfet_on scalars correspond to a rising input wave. The four equations below, one for each scalar type, control the time indexing of the scalars.

[0039] pfet_on_time = mod ((time + period -fall), period) - ((period * (1 - edge)) + rise - fall)

[0040] pfet_off_time = mod ((time + period -fall), period)

[0041] nfet_on_time = mod ((time + period -rise), period) - ((period * edge) - rise + fall)

[0042] nfet_off_time = mod ((time + period -rise), period)

[0043] where

[0044] time = simulation time parameter

[0045] period = time per cycle

[0046] rise = rising input edge arrival time

[0047] fall = falling input edge arrival time

[0048] edge = 0 if fall > rise and 1 if rise > fall

[0049] Other timing schemes can be used. They will likely have both advantages and disadvantages over the equations enumerated above.

[0050] While the "local times" are now periodic, they do not yet account for variations in input transition rate, supply voltage(s), or temperature. Sensitivity to these parameters is illustrated in Figure 3 for the NFET turn-on scalar. The waveform NFET turn-on base represents the NFET turn-on scalar as generated at the base input transition rate, supply voltage(s), and temperature. As these parameters vary, the scalar further varies as the waveform NFET turn-on; only the time scale is changing, because the scalar is already normalized to its' dc impedance. This time scale change is implemented by defining a "modified local time" with a piece wise-linear monotonic relationship to the original local time. The "modified local time" has a delay K7 and a change in slope K8, with parameter K9 initiating the slope change. The K7 through K9 parameters are defined from the timing points in Figure 3 according to the equations (2) below:

[0051] K7 = T2 - T0

[0052] K8 = [(T3 - T2) - (T1 - T0)] / (T1 - T0)

[0053] K9 = T2 (2)

[0054] We next use K7-K9 to define "modified local time" as in the equation (3) below.

[0055] NFET_on_time_mod = NFET_on_time + K7 +

[0056] + K8*[max(NFET_on_time-K9, 0)] (3)

[0057] When (NFET_on_time > K2) --> Adjust slope according to K1

[0058] Else --> No adjustment to slope

[0059] Like the D0 parameter in equation (1), values for K7, K8, and K9 are calculated prior to run time using empirical equations which are functions of temperature, supply voltages, and input transition rate obtained from actual parameter values. If conditions at run time are the same as those used to generate the nfet turn-on base waveform in Figure 3, then K7 and K8 are set to zero; otherwise, they are set to non-zero values to appropriately modify the time sequence parameter. The turn-on impedance of the NFET can now be represented as the product of the scalar (in this case the NFET turn-on scalar) which varies with "modified local time" and the NFET dc impedance.

[0060] To account for all these variations in each scalar, the scalar indexing equations listed above are modulated using the K1 - K12 terms below as a function of supply voltage(s), temperature, and input slew rate.

[0061] pfet_on_time_mod = pfet_on_time + (K1 + K2*(pfet_on_time -K3))

[0062] pfet_off_time_mod = pfet_off_time + (K4 + K5*(pfet_off_time - K6))

[0063] nfet_on_time_mod = nfet_on_time + (K7 + K8*(nfet_on_time - K9))

[0064] nfet_on_time_mod = nfet_off_time + (K10 + K11*(nfet_off_time - K12))

[0065] Where Kxx = f{ voltage, temperature, and input slew }

[0066] To account for dc fet impedance sensitivity to temperature and supply voltage changes, the dc impedance for each element in figure-2 is modulated using the K13-K16 terms below as a function of voltage and temperature.

[0067] power_clamp_dc_mod = power_clamp_dc * (1 + K13)

[0068] ground_clamp_dc_mod = groung_clamp_dc * (1 + K14)

[0069] pfet_dc_mod = pfet_dc * (1 + K15)

[0070] nfet_dc_mod = nfet_dc * (1 + K16)

[0071] The nfet and pfet impedance is given as

[0072] Rpfet = [pfet_dc_mod]*[pfet_on_scalar] || [pfet_dc_mod]*[pfet_off_scalar]

[0073] Rnfet = [nfet_dc_mod]*[nfet_on_scalar] || [nfet_dc_mod]*[pfet_off_scalar]

[0074] where

[0075] pfet_on_scalar = f{pfet_on_time_mod}

[0076] nfet_on_scalar = f{nfet_on_time_mod}

[0077] pfet_off_scalar = f{pfet_off_time_mod}

[0078] nfet_off_scalar = f{nfet_off_time_mod}

[0079] The power and ground clamp impedance is given as

[0080] Rpwrclamp = power_clamp_dc_mod

[0081] Rgndclamp = ground_clamp_dc_mod

[0082] Again it should be emphasized that the use of PFETs and NFETS is just illustrative. The model can be used with any switching element that pulls up or pulls down voltage.

[0083] Alternatively, the model can use conductance versus impedance when creating the model. FIGS 4a, 4b, and 4c illustrate this. Example of variable conductance as a function of device voltage and time.

[0084] FIG 4a illustrates how the DC conductance of a device is obtained. The DC conductance of a device (e.g. Pullup or Pulldown) is characterized as a function of device voltage. For this example, the conductance varies from 10-35 over a device voltage range of 0-3 volts.

[0085] FIG 4b illustrates how the transient conductance of a device is obtained. To capture the transient conductance of the device, the DC conductance of the device is multiplied by a scalar that is a function of time. For this example the scalar ranges from 1, corresponding to when the device is turned on, to 0 when the device is turned-off. The scalar represents the rate at which the device turns off.

[0086] FIG. 4c illustrates the transient conductance of a device. The transient conductance of the device is shown to vary from 35, corresponding to when the device is on, to 0 when the device is off. Note that the transient conductance is a function of both device voltage and time. Note that time is referenced to the midpoint of the input signal. This synchronizes the input and output waveforms of the BIO model.

[0087] A simple example to illustrate the concept of a BIO model adjustment parameter based on conductance scalars and how it is handled within the BIO model is presented in Figures 5a, 5b and c. Figure 5a illustrates the determination scalar values as a function of time at some fixed voltage and temperature denoted by Env1. Figure 5b illustrates how the scalars values from Figure 5b are represented in tabular format within the BIO model. Figure 5c is simple illustration of how these scalar values might change due to variations in voltage, temperature, and input slew rate denoted by Env2.

[0088] The goal is to be able to use the scalar table (Scalar@Env1) within the BIO model to accurately reproduce the scalar values at Env2. To do this, we introduce a time shift parameter K0 such that ...

[0089] Scalar@Env2(Time) = Scalar@Env1(Time-K0) where K0=1 for this example

[0090] K0 = f{voltage, temperature, input slew rate}

[0091] With the appropriate value for K0 calculated prior to simulation, the scalar table within the BIO model can be adjusted for any time shift due to changes in voltage, temperature, and input slew rate.

[0092] Figure 6 illustrates the behavioral model topology. The output drive stage consists of resistances which are functions of input edge arrival times, input transition rate, supply voltage(s), device voltage, and temperature representing the NFET and PFET as described above. The ESD stage contains resistances representing the power and ground ESD structures which are functions of supply voltage(s), device voltage, and temperature.

[0093] While the output drive and ESD clamp stages contain the behavioral characteristics necessary for timing and signal integrity analysis, the pre-drive current and decoupling stages contain additional behavioral characteristics necessary for noise and power grid integrity analysis. Each device within these stages can be represented by either a fixed-value element, a non-switching element that is a function of parameters not varying in time, or a switching element which is a function of both time and non-time varying parameters. A tradeoff between model complexity and accuracy typically defines the type of element used.

[0094] There are many characterization techniques that we have developed for generating the data that describes the elements of the model. The I/O family type determines the appropriate techniques to use. Note that this only affects the data values that go into the model, not the model topology or algorithms. Figure 11 illustrates one of the characterization techniques used for the decoupling stage. The method for characterizing the decoupling structure of Figure 11 involves the following steps:

[0095] 1) Switch S1 is closed at t_o and charges node A to V_{cc}.

[0096] 2) Switch S1 is open before t_1.

[0097] 3) Switch S2 is closed at t_1.

[0098] 4) From the waveform:

[0099] Cnear = $(V2*Cext)/(V_{cc} - V1)$

[0100] Cfar = $(V2*(Cnear=Cext)-V_{cc}*Cnear)/(V_{cc}-V2)$

[0101] R_{pi} is calculated from the RC time constant associated with the waveform going from V1 to V2 where

[0102] C=Cext+Cfar+Cnear and R=R_{pi}.

[0103] The I/O family type is also used to characterize the predrive stage. For this characterization, the supply current minus the output stage current is used to define the resistance (conductance) of the predrive in the equation as follows:

[0104] R1(t) = Vcc1/Icc1 and R2(t) = Vcc2/(Icc2-output stage current)

[0105] R1 and R2 are then recorded in tabular format within the model.

[0106] Typically there is good output waveform correlation among the "BIO" model of Figure 6, IBIS models and full netlist models. Model accuracy begins to diverge after a change in voltage Vcc1 to the pre-drive circuits, temperature, and input transition rate. While the BIO model and full netlist models track across a wide range of conditions, the IBIS model is unable to adjust for varying conditions. When the output supply voltage Vcc2 is also allowed to change, IBIS model inaccuracies typically increase further. The high level of correlation between the BIO model and the full-netlist model permits BIO models use in both timing and signal integrity analysis. For a given I/O the same BIO model can be used across a wide range of conditions for signal integrity analysis, unlike an IBIS model that is only valid at a fixed voltage, temperature, and input slew rate. This, together with the BIO model's ability to capture driver delay and power grid current waveforms, makes BIO models much more versatile and suitable for an ASIC environment.

[0107]

The equation overhead and six resistive element BIO model (2 static elements, 2 pfet elements, and 2 nfet elements) shown in Figure 6 exhibit outstanding accuracy across a wide range of voltage, temperature, input slew rate, and off-chip net conditions and has been demonstrated across a wide range of I/O family types (e.g. CMOS, PECL, LVDS, GTL, etc,.) In addition whether being used for timing, noise, or signal integrity analysis, the BIO model's simulation times are typically less than 1% of the simulation time required for full-netlist models. The difference in simulation time becomes even more pronounced for increasingly complex I/O's. The reason is because the complexity of the behavioral model remains essentially constant as the complexity or size of the full netlist model increases. This makes possible a merger of analog and digital simulation techniques at the chip level.

[0108] Using the behavioral model of Figure 6, the static timer can merge on-chip and off-chip timing, perform signal integrity and perform I/O power grid integrity analysis as illustrated in Figure 7. The scope of internal chip static timing has historically ended at the silicon-package boundary. At this boundary various techniques have been used to manually interface internal chip timing with off-chip timing that trade accuracy with simulation run time. The most accurate technique is to use a static timer to determine edge arrival times at the input of the driver and then in a separate spice run that uses full spice netlist representations of the driver and off-chip net, determine delay from the input of the driver to the sink end of the net. The advantages of this approach are accurate off chip timing and the ability to perform signal integrity analysis. The disadvantages are 1) excessive run times when simulating full spice netlist I/O models that prohibit this technique from being applied at the chip level timing, 2) manual effort is required to merge on-chip timing with off-chip timing, and 3) many customers do not have the required software licenses for simulating encrypted spice models.

[0109] A technique that trades accuracy for faster run time is to calculate a lumped load capacitance that represents the effective capacitance as seen by the driver and use this lumped load in an empirical timing model such as a NDR (New Delay Rule) for calculating driver delay during static timing. A spice run is then required to calculate off-chip net delay from the output of the driver to the sink end of the net. The advantages of this approach are fast run time and reasonable driver delay accuracy assuming the effective capacitance was calculated correctly. The disadvantages of this approach are 1) the actual shape of the waveform at the output of the driver is not captured, therefore the input stimulus used during spice simulation of the network is incorrect and will affect accuracy, 2) this technique does not support signal integrity analysis, and 3) manual effort is required to merge on-chip and off-chip timing.

[0110] This invention provides one with the ability to embed spice behavioral models within a static timer such as EinsTimer licensed by IBM or other industry available static timers like PrimeTime by Synopsys and BuildGates by Cadence that preserve the accuracy of a full netlist model, run orders of magnitude faster, and gives the static timer the ability to do signal integrity and I/O power grid analysis with the accuracy of spice at the chip level. A static timer is used to time the internal logic up to the input of the driver in order to determine both the edge arrival time and slew rate to the driver. A spice deck builder within the static timer is used to build a spice deck that references both the driver's spice behavioral model and the spice netlist representing all parasitic and transmission line characteristics from the output of the driver to the sink end of the net. User defined measurement criteria such as voltages thresholds for delay measurements, frequency, number of cycles, and waveform observation points would be input to the deck builder that would control delay measurement and signal integrity analysis. Using the spice deck and the information of edge arrival time and slew rate to the driver, the static timer then invokes spice (e.g. PowerSpice) and simulates the entire net in a small fraction of the time as compared to simulating the entire net using full spice netlist I/O models. The spice run results are merged with static timing assertions by the static timer to form a seamless timing interface between on-chip and off-chip timing.

[0111]

Figure 7 illustrates how such a static timer can be combined with the BIO models to produce these results. Primary input (PI) assertions are used to define edge arrival time and slew rate at the input pins of the chip. The static timer uses the PI assertions (100), a chip level cell netlist, and on-chip parasitic information to perform a customary static timing analysis (102). The results of this analysis are available in the internal timing reports (107) that describes all path timing internal to the chip. Measurement criteria for off-chip timing and signal integrity analysis are defined (101) and used as input to the spice deck builder (103) within the static timer. The spice deck builder creates a spice deck (104) that references the off-chip spice netlist (105) and I/O behavioral model (106). Using the spice deck and driver input assertions from (102), the static timer invokes spice to simulate the off-chip net. From the spice simulation results, the static timer produces off-chip timing reports (108) for delay analysis, off-chip voltage waveforms (109) for signal integrity analysis, and I/O power grid integrity reports (110) using the current waveforms from the spice simulations and cell level decoupling information in the NDRs.

[0112] This invention also provides a mixed signal simulation environment using the BIO models. Figure 8 illustrates the use of the BIO model in the mixed signal domain. The mixed signal simulation back plane consists of three basic components; a digital simulation domain 20, for example an NDR rule containing empirical equations for calculating gate delay, an analog simulation domain such as a SPICE or PowerSPICE simulation engine 30, and a control module 10 that merges the two simulation domains during analysis. All three elements are typically coded within the same software language environment (e.g., DCL ("Delay Calculator Language")) that allows the simulation back plane to be ported across multiple operating systems such as RS6000 AIX, Solaris, HP, and Linux. By definition of a back plane, this mixed signal simulation capability can be exploited by multiple applications. One example is accurate on-chip/off-chip timing within a static timer like that shown above using BIO models.

0968354б.011602

[0113] Step1 is to determine I/O driver delay. To accurately determine I/O driver delay, a static timer such as EinsTimer licensed by IBM or other industry available static timers like PrimeTime by Synopsys and BuildGates by Cadence passes the simulation conditions to the control module 10. These conditions specify the level of modeling required to get the desired level of accuracy. The basic choices are 1) analog such as full device SPICE net list models, 2) combination of analog-digital such as BIO models, and 3) digital such as NDRs. For this example analog-digital simulation techniques are chosen as the optimal balance between simulation speed and accuracy. The simulation conditions also specify the rail voltages, temperature, process corner, input-output edge pair, and input slew rate to the driver.

[0114] Step 2 is to supply a SPICE deck. If a SPICE simulation deck 60 does not exist, the control module 10 builds one using the appropriate BIO model 40 and output load 50. The output load describes all silicon, package, and board level parasitics the driver will see.

[0115] In Step 3 the control module 10 passes the environmental conditions in the parameter library 20. The environmental conditions being obtained from the simulation condition in the control module.

[0116] In Step 4 the parameter library 20 passes BIO model adjustment parameters back to the control module 10 based on the environmental conditions.

[0117] In Step 5 the control module (1) passes the adjustment parameters, environment conditions, BIO Parameter and SPICE deck (6) into the SPICE simulation engine (3). Note at this step the analog characteristics of the BIO model have been dynamically adjusted using digital simulation techniques prior to analog simulation

[0118] The final step, Step 6, is the spice SPICE simulation. Results are then passed back to EinsTimer to be included in static timing reports.

[0119] Another application that this mixed-signal simulation provides is the ability for greater accuracy on critical paths. Critical paths could be determined automatically from slack reports or manually via user input. Circuit partitioning for leveraging various simulation engines in a seamless interface is a natural attribute for this system.

[0120] Furthermore, the present invention may be implemented in an information handling/ computer system. For example, Figure 9 illustrates a typical hardware configuration of an information handling/computer system for modeling integrated circuits in accordance with the invention.

[0121] As shown in Figure 9, the inventive information handling/computer system (400) preferably has at least one processor or central processing unit (CPU) 411. The CPUs 411 are interconnected via a system bus 412 to a random access memory (RAM) 414, read-only memory (ROM) 416, input/output (I/O) adapter 418 (for connecting peripheral devices such as disk units 421 and tape drives 440 to the bus 412), user interface adapter 422 (for connecting a keyboard 424, mouse 426, speaker 428, microphone 432, and/or other user interface device to the bus 412), a communication adapter 434 for connecting an information handling system to a data processing network, the Internet, an Intranet, a personal area network (PAN), etc., and a display adapter 436 for connecting the bus 412 to a display device 438 and/or printer 439 (e.g., a digital printer or the like).

[0122] In addition to the hardware/software environment described above, a different aspect of the invention includes a computer-implemented method for performing the above method. As an example, this method may be implemented in the particular environment discussed above.

[0123] Such a method may be implemented, for example, by operating a computer, as embodied by a digital data processing apparatus, to execute a sequence of machine-readable instructions. These instructions may reside in various types of signal-bearing media.

[0124] Thus, this aspect of the present invention is directed to a programmed product, comprising signal-bearing media tangibly embodying a program of machine-readable instructions executable by a digital data processor incorporating the CPU 411 and hardware above, to perform the method of the invention.

[0125] This signal-bearing media may include, for example, a RAM contained within the CPU 411, as represented by the fast-access storage for example. Alternatively, the instructions may be contained in another signal-bearing media, such as a magnetic data storage diskette 500 (Figure 10), directly or indirectly accessible by the CPU 411.

[0126] Whether contained in the diskette 500, the computer/CPU 411, or elsewhere, the instructions may be stored on a variety of machine-readable data storage media, such as DASD storage (e.g., a conventional "hard drive" or a RAID array), magnetic tape, electronic read-only memory (e.g., ROM, EPROM, or EEPROM), an optical storage device (e.g. CD-ROM, WORM, DVD, digital optical tape, etc.), paper "punch" cards, or other suitable signal-bearing media including transmission media such as digital and analog and communication links and wireless. In an illustrative embodiment of the invention, the machine-readable instructions may comprise software object code, compiled from a language such as "C", etc.

[0127] While the invention has been described in terms of specific embodiments, it is evident in view of the foregoing description that numberous alternatives, modifications and variations will be apparent to those skilled in the art. Thus, the invention is intended to encompass all such alternatives, modifications and variations which fall within the scope and spirit of the invention and the appended claims.